TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TUOMAS VAHERKOSKI

ESTIMATING PERFORMANCE OF AUTOMATED GUIDED VEHICLE SYSTEMS

Master of Science thesis

Examiner: Prof. Kari Systä
Examiner and topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on 17th August 2016

ABSTRACT

TUOMAS VAHERKOSKI: Estimating performance of automated guided vehicle systems
Tampere University of Technology
Master of Science thesis, 41 pages
November 2017
Master's Degree Programme in Information Technology
Major: Pervasive Systems
Examiner: Prof. Kari Systä
Keywords: Automated guided vehicle, performance, simulation

The use of automated guided vehicle (AGV) systems is increasing rapidly in warehouse and manufacturing environments. However, selling and commissioning AGV systems have certain challenges. Especially estimating AGV system performance in early stages of the sales process requires a lot of time and skill. Estimating system performance is important because selling the system for too cheap would not be financially sustainable. Also, a pricey offer will improbably make the deal.

AGV system performance can be estimated using simulation tools. By running the simulation with a different number of vehicles and measuring the throughput, one can decide how many vehicles should be used. Previously Rocla has used a simulator that is slow and difficult to use in the sales process. This Master of Science thesis will introduce a new simulation method that is based on time-window based routing algorithm developed at Rocla.

The new simulator produces an estimate of AGV system performance with different vehicle counts. A proof of concept implementation of the new simulation method was developed to evaluate its accuracy and runtime performance. Its correctness was verified using visualization of the simulation. When comparing the new simulator to the previously used one, the results seem very promising. Both simulator produce similar estimate of the system performance. However, the new method performs significantly better in terms of runtime performance. It cuts the simulation time to less than a minute while previously used method takes multiple hours.

TIIVISTELMÄ

TUOMAS VAHERKOSKI: Automaattitrukkijärjestelmien suorituskyvyn arviointi
Tampereen teknillinen yliopisto
Diplomityö, 41 sivua
Marraskuu 2017
Tietotekniikan koulutusohjelma
Pääaine: Pervasive Systems
Tarkastaja: Prof. Kari Systä
Avainsanat: Automaattitrukki, suorituskyky, simulaatio

Automaattitrukkijärjestelmien käyttö varastoissa ja tehtaissa yleistyy jatkuvasti. Uusien järjestelmien myynnissä ja käyttöönotossa on kuitenkin omat haasteensa, sillä järjestelmän suorituskyvyn arviointi vaatii paljon teknistä osaamista ja aikaa. Suorituskykyä pyritään arvioimaan mahdollisimman tarkasti, jotta asiakkaalle tehtävä tarjous olisi mahdollisimman kilpailukykyinen.

Järjestelmän suorituskykyä voidaan arvioida muun muassa simuloimalla järjestelmää eri trukkimäärillä, ja mittaamalla kuinka tehokkaasti se pystyy suorittamaan annetut tehtävät. Aiemmin Roclan käytössä ollut simulaatiomenetelmä ei kuitenkaan toimi riittävän nopeasti ja helppokäyttöisesti, minkä takia sen käyttö myynnin työkaluna on haastavaa. Ongelman ratkaisemiseksi tässä diplomityössä on kehitetty simulaatiomenetelmä, joka perustuu Roclan kehittämään aikaikkunapohjaiseen reitinhakualgoritmiin.

Kehitetystä simulaatiomenetelmästä tehtiin prototyyppitoteutus, jonka avulla menetelmän toimivuutta lähdettiin arvioimaan. Simulaation tuloksena saadaan arvio tehtävämäärästä, jonka automaattitrukit pystyvät suorittamaan. Simulaation toimivuuden varmistamiseksi se voidaan myös visualisoida. Visualisaation avulla saadaan myös osoitettua järjestelmän toimintatapa uudelle asiakkaalle.

Verrattaessa kehitetyn simulaattorin antamia tuloksia aikaisemmin käytössä olleeseen menetelmään, havaittiin, että tulokset vastaavat hyvin toisiaan. Kehitetyn simulaattorin avulla järjestelmän suorituskyky voidaan arvioida minuuteissa, entisen, useita tunteja kestävän prosessin sijaan.

PREFACE

This Master of Science thesis was written for Atostek Oy and Rocla Oy. I would like to thank both of them for providing the topic of this thesis and for funding the writing of it.

I would also like to thank all of my instructors, Professor Tommi Mikkonen, Professor Kari Systä and Petteri Vigren. They all have tremendously helped the thesis writing process by giving helpful feedback and constructive comments about the work.

Additionally, I want to thank my work team members, Juho, Jani and Anssi, who have provided useful insights about automated guided vehicle systems. Also thanks to my family and friends for all the support they have provided me.

In Tampere, Finland on 24th of October 2017

Tuomas Vaherkoski

TABLE OF CONTENTS

ABBREVIATIONS

AGV	Automated Guided Vehicle
CPU	Central Processing Unit
CS	Continuous simulation
CSV	Comma-separated values
DES	Discrete-event simulation
FCFS	First-Come-First-Served
ID	Identification number
JSON	JavaScript Object Notation
PoC	Proof of Concept
SPPTW	Shortest Path Problem with Time-Windows
SQL	Structured Query Language

1. INTRODUCTION

As the world is becoming increasingly automated, there is need to develop control software for automation systems. Rocla Oy with Atostek Oy has been developing a control and routing system for Automated Guided Vehicles (AGVs). The control system manages and optimizes the movement of vehicles in a fully or partially automated warehouse. In such warehouses each vehicle has transportation orders with a start and destination location. The vehicles are moving in shared space, so the system must be careful not to drive them into positions where they cannot move (deadlock) which requires careful planning in time and space domains. Because old AGV control systems often left the management of traffic rules to the system designer, a new control system was developed at Rocla. Its main advantage is easier and more flexible design process which does not require manual configuration of traffic rules.

The project for developing the new control system started in early 2014 and the first version of the product was released in 2016. At the same time as first production versions were released and deployed, there came an idea about developing improved methods for estimating the performance of the new control system. The idea emerged from a continuous need to estimate system performance in the sales and design phases of new AGV systems.

Selling complex automated guided vehicle systems to end-users requires a great deal of case-specific knowledge. Currently there is no quick, easy and accurate method for estimating the number of required vehicles for desired system throughput. Simulating a full system is quite expensive and time-consuming, so often the sales must rely on good guess and acquired experience rather than measured or calculated data. This uncertainty may lead unnecessarily high safety margins in the sales offers for new customers. Especially in the early stages of the sales process, it would be a huge benefit if the sales team could make accurate predictions about the system performance with a certain number of vehicles.

This thesis will introduce the problems in AGV system design and operation. It will then propose a new method for estimating the system performance using a much faster simulation method than what was already in use at Rocla. The new simulation method is designed specifically for the new AGV control and routing system developed by Rocla. The main functionality of the new simulator is to calculate best possible estimate for the performance of the new control system. The reason for me to choose this topic was my involvement at Atostek in the design and implementation of the new control system. Also the subject was very interesting and I had a good idea how the simulator should be implemented.

The second chapter of this thesis will guide the reader to problems in AGV system design and operation. It will then describe the performance measures commonly used for measuring the performance of AGV systems. Finally the different methods of performance analysis are introduced with their strengths and weaknesses.

The new simulation method is introduced in chapter three. The chapter starts by introducing the routing algorithm that lays the foundation for the new simulation method. Then the input requirements for valid outputs are defined. Finally the outputs of the new simulation method are explained and their properties are discussed.

The fourth chapter contains a technical description about the proof of concept implementation for the new simulator. It tells about technologies which are used in the implementation, describes the software architecture and the data flow inside the program and gives a detailed description how the new simulator works internally. The chapter also tells how a user can interact with the simulator.

The fifth chapter compares the new simulation method to the old one. The comparison is done by running a test system simulation using both simulation methods. Calculated performance metrics are then compared and the differences are discussed. Chapter five also discusses about extendability and usability aspects of developed simulator application. Additionally possible improvement areas are considered and implementation and usage experiences are discussed.

2. SYSTEM DESIGN PROCESS

According to Le-Anh and Koster [1] there are multiple key issues in design and operation of automated guided vehicle systems. The design problems include guide-path design and choosing the number of vehicles in the system. Usually the number of required vehicles is chosen based on the system throughput requirements. Operational problems in AGV systems are order scheduling, vehicle routing, idle vehicle placement, battery management and deadlock resolution. Vis [2] also lists failure management as an issue. These problems are further discussed and explained in sections 2.2 and 2.3.

2.1 Automated Guided Vehicles

An automated guided vehicle is a robot vehicle that navigates autonomously without a human driver. They are widely used in factory or warehouse operations for various tasks such as automated order picking or block storage [3]. AGVs have a few obvious advantages over human-operated vehicles. Firstly as they do not require human drivers, money can be saved in salaries. Secondly, AGVs are excellent in executing basic transfer tasks which human drivers may find boring and repetitive. It has been observed that AGVs also increase the safety on work premises. An example how an AGV may look like is shown in figure 2.1. The image shows one of the AGV models sold by Rocla which is designed for standard pallet handling and transferring operations.

2.2 Design problems

The design process of an AGV system involves various steps. It starts by requirement gathering, followed by design phase and finally analyzation whether the requirements are met. As in software development, the design process of new AGV systems has become more agile over the years. More flexible technologies and tools have been

Figure 2.1 *Rocla Fork Over Lift AGV [3]*

adopted. This has enabled iterative development process where the system engineer can experiment, test and revise the design based on the results of analyzation.

One of the first problems in AGV system design process is deciding what kind of guide-path design is chosen [1]. A guide-path consist of points and segments and can be thought as a directed graph [2]. The purpose of guide-path is to define the routes which the vehicles can use for driving around the warehouse. Previously vehicles used magnetic strips on the floor for navigation, but they have been mostly replaced by laser scanners [1] and other modern technologies. Modern navigation methods allow the guide-path layout to be changed via software update for the vehicle and controller system, which means that the system is not tied to a fixed layout after the initial construction.

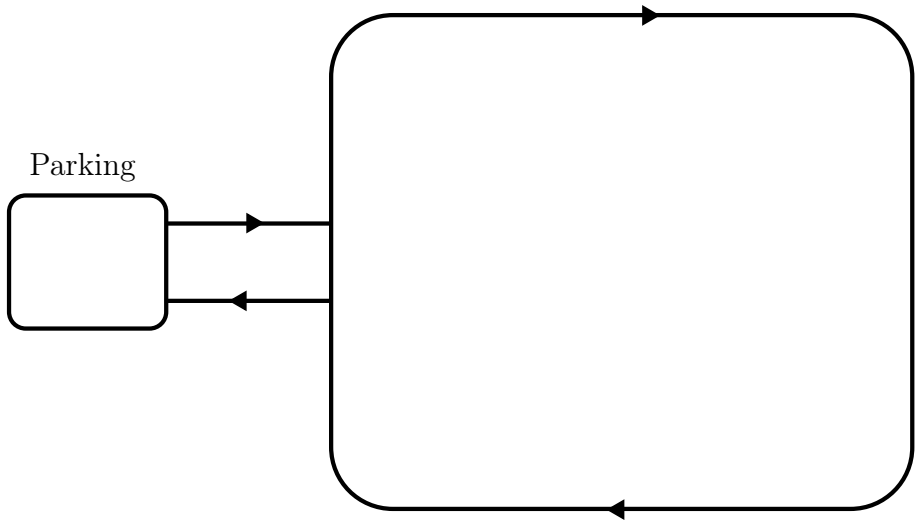

***Figure** 2.2 Loop layout*

There are various commonly used guide-path topologies, and making the choice between them can be difficult due to lack of available information and requirements of the system. Le-Anh and Koster [1] categorize topologies in three different groups: conventional, single loop and tandem topologies. Conventional topology, seen in figure 5.1, is the most flexible but also the most complex topology. It is basically a mesh network with arbitrarily complex connections between nodes. Pick-up and drop-off stations of loads moved by vehicles can be placed anywhere in the network. In single loop topology seen in figure 2.2 there is only a single loop in which there are no junctions or shortcuts. Pick-up and drop-off stations are placed along the loop. Loop layouts are mainly used in small systems because the traffic control and routing are easy to implement. The third, tandem topology seen in figure 2.3, consists of multiple separate areas which do not interfere with each other. Only single vehicle serves each area and transfer stations are used for moving loads from one area to another. In addition to topology, other characteristics of a guide-path design are the number of parallel lanes and flow direction. Use of parallel lanes may increase the system performance if there is enough room for vehicles to bypass each other. There should be enough possibilities for changing lanes if currently used one is blocked. The guide-path flow direction is either unidirectional or bidirectional. Bidirectional path allows the vehicle to use the same aisle for driving in both directions. This is beneficial especially in cases where there is no room for multiple parallel lanes.

Creating a good guide-path design requires expertise and time. In addition to choosing the guide-path topology, the number of parallel lanes and flow direction,

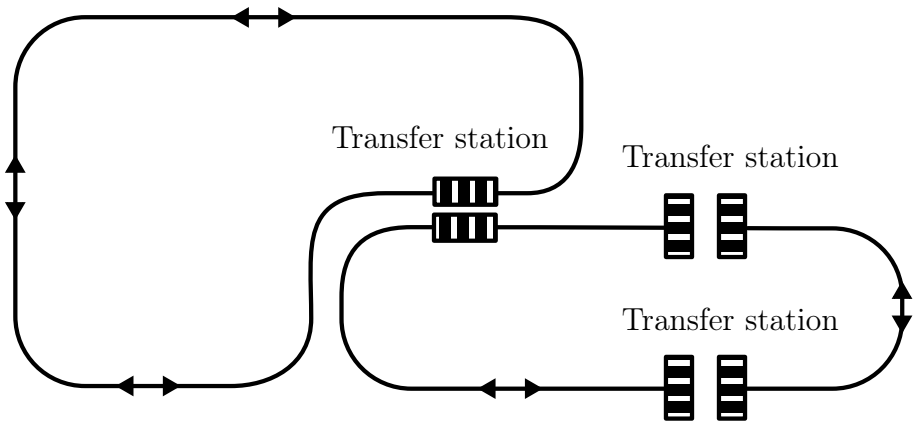

Figure 2.3 *A tandem layout with three zones [1]*

the designer may have to consider optimal locations for different stations. Pick-up, drop-off and charging stations are usually in fixed locations in the warehouse, while idle locations can be selected more freely. However, idle vehicle placement problem is difficult because the locations of pick-up orders usually are not known beforehand. By placing the idle vehicles close to pick-up stations, the drive time of empty vehicles is reduced, which improves the efficiency of the system. The designer should choose the idle locations so that they disrupt normal vehicle flow as little as possible. This can be challenging if there is not enough room to place the idle locations off drive paths. The environment may also have various dynamic obstacles such as some factory process which occasionally blocks the drive path. Because of this, multiple alternative routes may be needed for bypassing.

According to Le-Anh and Koster [1] conventional bidirectional guide-path systems are not popular in material handling systems due to their complexity. However, the conventional layout has many advantages such as flexible routing (multiple alternative routes), shorter travel distances and tolerance to system failures. Disadvantages are complicated control, congestion and interference problems and difficulty of expansion. If these problems can be resolved, the conventional layout is a good choice in any environment.

2.3 Operational problems

Operation of large fleets of AGVs in highly dynamic warehouse environment requires a great deal of control. The schedules are constantly changing as new orders are placed, factory machines complete their work, vehicles break or some manually operated machine blocks the way of a vehicle. Order scheduling is the problem of deciding when a specific order should be executed, which vehicle should execute the order and what route it should use [1]. According to Qiu, Hsu, Huang, *et al.* [4], the intent of order scheduling is to dispatch a fleet of vehicles to perform queued orders while achieving specified constraints. These constraints are usually related to system performance requirements. For example, a certain amount of orders must be completed during a work shift. Some orders may also have other orders as dependencies. Vis [2] argues that to increase performance, large systems should be able to reschedule already scheduled orders due to large distances between a source and a target location of vehicles.

The problem of selecting a vehicle that should execute a transportation order is called the dispatching problem [2]. The problem can be approached in two different ways. In vehicle initiated dispatching the vehicle that has completed its previous order must select a new load from a set of requested items. Alternatively, in work center initiated dispatching the system must select a vehicle for an order from a set of idle vehicles.

There are multiple factors that should be considered in order dispatching process. Some of these are route constraints (does a route exist between the vehicle and the target station), vehicle type constraints (can the vehicle handle the type of load), empty driving time and distance, and battery status. In a warehouse environment, the system often has to make dispatching decisions online as new orders are received unpredictably. Dispatching can happen off-line if all orders and transportation times are known in advance. However, in such case, one must expect no failures or other unpredictable events to occur. There is also a case where no new orders are waiting to be executed when a vehicle completes its previous order. Such vehicle should be routed to an idle location where it can wait for new orders to arrive.

Once an order has been dispatched to a vehicle, the system must decide which route the vehicle should take in order to reach its destination. There are two different approaches [2] to the routing problem: static and dynamic. In static routing, the route from point a to point b has been predetermined (usually the shortest path)

and is always used when new order from a to b is requested.

The problem with static routing approach is that it can not optimally handle vehicle breakdowns or other unexpected events such as human intervention. In static approaches the conflict avoidance is often implemented using traffic rules such as one-way paths or single vehicle per zone rules. This leads to sub-optimal utilization of available paths. Without traffic rules the system may run into a deadlock. A deadlock is a situation where more than one vehicle is trying to drive to the same location at the same time. This means that the vehicles are waiting for each other to move away from that location and the system becomes locked. Deadlock situations must be resolved either automatically or manually by the system operator.

Traffic rules may also require changes and adjustments when a system vehicle count is changed. The rules can also be hard to verify and they require extensive testing so that the designer can be sure they work as intended.

Dynamic routing strategy is a counterpart for static routing and can be used to overcome the problems in static routing. The dynamic routes are not predetermined which allows the system to calculate the route based on the information about the current traffic situation. A few previously published papers discuss the Shortest Path Problem with Time-Windows (SPPTW) [5], [6]. The idea is to construct a conflict-free route for a single vehicle. The constructed plan is then reserved globally so that the following routings must respect the reservations of the routed vehicle. The reservations are released for use as the vehicle drives along the planned route.

After the route has been decided, the vehicle should start moving. Considering that many unexpected events might occur after the route has been decided, the system should be ready to account for delays in the routes. The control of vehicle movements is usually implemented using centralized traffic controller. The controller orchestrates the vehicle fleet by giving them drive commands and solving or avoiding deadlock situations. Previous research seems to ignore the possibility of deadlock after the vehicle has reached its destination. The method represented in this thesis will solve this problem by reserving an additional route from the target station to a location where the vehicle can wait until a free path opens again.

The purpose of different routing algorithms is to optimize different factors of the routing process. One routing algorithm may provide less optimal routes but perform better resource wise (CPU time, memory). On the other hand, some algorithms may

seek better or the best possible route in terms of vehicle travel distance or drive time. Scheduling may also impose some limitations or rules for the routing algorithm. Selected routing algorithm can also restrict the layout topologies available for the designers. For example, a loop layouts can be routed using a simple static routing algorithm [4].

Finally, the battery management of vehicles is required if they use rechargeable batteries. The responsibility of battery manager is to keep the system operational in conditions where there are so many orders that all available vehicles are needed. Battery management strategies vary from simple "battery empty, go to charge" to more complex and adaptive strategies where the energy level of a system is constantly kept above some threshold level.

2.4 Performance measures

Beamon [7] comprehensively reviews the performance measures used in the design and analysis of AGV systems. Previous research has studied the AGV system performance as a function of layout design, fleet size and vehicle properties such as speed and load capacity. Operative properties of an AGV system such as vehicle dispatching and order scheduling rules, positioning of idle vehicles, routing and congestion avoidance techniques have also been seen as factors in the AGV system performance.

Beamon categorizes the performance measures into six areas: quality, time, cost, resource utilization, operating efficiency and flexibility measures. In this thesis, the focus is on time and operating efficiency measures. One of the obvious throughput metrics is the number of completed orders per time unit. This throughput metric does not usually increase linearly when more vehicles are added to the system. A simple reason is the path congestion which causes delays to the AGV system as other vehicles must wait for each other.

According to [2] there are following objectives for an AGV system: maximize throughput of the system, minimize the time required to complete all orders, minimize vehicle travel times, evenly distribute workload over AGVs, minimize total costs of movement, minimize the time required to travel from idle location to pick-up stations, minimize expected waiting times of loads. When these objectives are optimized perfectly the system runs very efficiently.

2.5 Performance analysis

The design choices have various effects on the performance of AGV systems. The goal of this thesis is to introduce a method for estimating the performance and throughput of AGV systems. This information can be used to analyze the effects of various design choices and it helps the designers in finding the most optimal design parameters for AGV systems. Finding correct design parameters for a new AGV system is not an easy task because the interactions of different choices might be unexpected and hard to foresee [2].

Estimating the performance of an AGV system is an important tool when assessing the required vehicle count. The customer might require some minimum throughput capacity, so a method for calculating how many vehicles are required is needed. Sales teams need to know the systems requirements when they make offers to the customers. The sales try to keep the vehicle count in minimum because if the required performance can be achieved using fewer vehicles, it is easier to compete with the cheaper total price of the system. The price of the vehicles plays a major role in the pricing of the system, and so, only one additional vehicle can make or break the deal. The sales would greatly benefit from tools that estimate the performance of the AVG system. However, they usually rely on experience and common sense rather than measured data. Although it is often possible to increase the system performance by simply adding more vehicles, it may be too big a risk to make an offer without margin for estimation error if too few vehicles are chosen.

There are two different approaches to performance analysis [2]. A simulation approach tries to gather data from a simulated system and calculates the performance metrics from the data. The second, analytic approach, tries to analyze the effects of design decisions to system performance using mathematical formulas and models. Next two sections introduce these approaches and highlight their strengths and weaknesses.

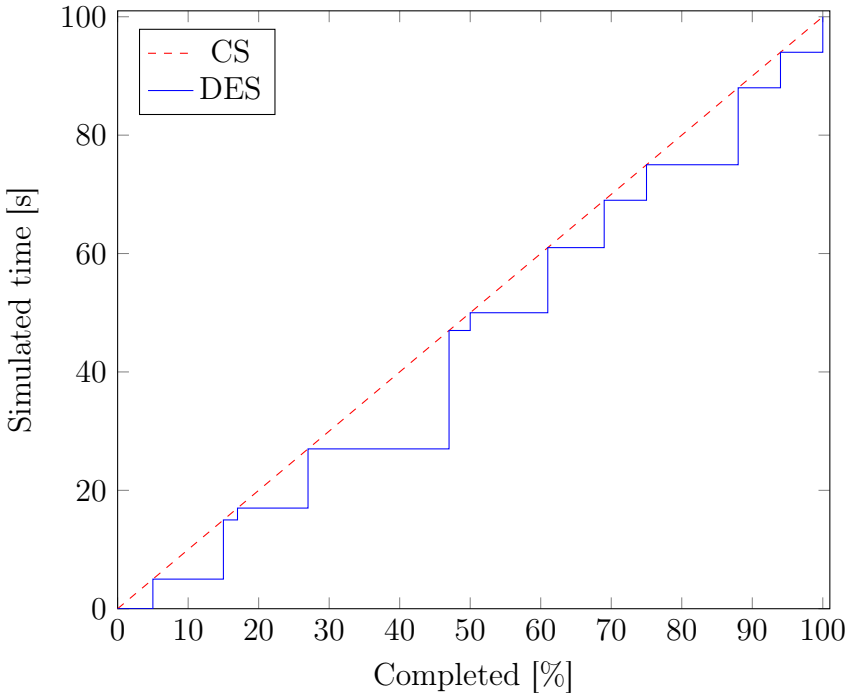

Figure 2.4 *Time in continuous and discrete-event simulation*

2.5.1 Simulation methods

Banks [8] defines the simulation as "the imitation of the operation of a real-world process or system over time". In the context of this thesis, a simulator can be used to run experiments where a set of orders are fed into a virtual AGV systems with varying vehicle counts. The simulator records timestamps of vehicle movements and state changes for later analysis. Obtained data can later be used for calculating the performance metrics.

Multiple simulation models have been designed for modeling AGV systems [2]. Using these simulation models and tools, a system engineer can create a simulation of AGV system. However, the cost of setting up and running the simulation can be expensive because the engineer may first have to design a layout, configure traffic rules to prevent the system from running into deadlocks, perform the simulations and finally analyze the results. The system configuration may require alterations based on the number of vehicles in the system which further increases the design cost. Simulation-based solutions have been researched, but they are considered a slow and costly solution. Most of them also model congestion only partially.

The simulators are traditionally categorized to continuous (CS) and discrete-event

simulators (DES) [8]. The difference between continuous simulation and discrete-event simulation techniques is quite noticeable. In a continuous simulation model, the system state changes gradually as time passes. For example, the continuous simulation model would model how a vehicle drives along its pathway and finally reaches its destination. In discrete-event simulation, the state of simulated system changes at discrete points in time. These points in time are called event times. In AGV system these event times are the arrival and depart from operation stations. The difference in the passage of time is illustrated in figure 2.4.

Currently Rocla uses a simulator that is implemented using continuous simulation technique. This simulator has problems when the user wants to simulate a system with varying number of vehicles. Running the simulations can take multiple hours or even days because accurate simulation is limited to real-time (1x) speed. This makes currently used simulator impractical for fast iteration. Additionally, only one simulation can be running at a time, so multiple PCs are needed to parallelize simulations with different configurations.

Rocla with Atostek has been developing a new product that allows scalability and easily configurable system layout without topological layout restrictions. The new solution provided by Rocla has solved many of the problems that are considered disadvantages for the conventional layout. This thesis will primarily focus on the estimation of system performance by introducing a method which uses discrete-event simulation method and allows accurate estimation of blocking effects in systems with large vehicle fleets on conventional layout topology with multiple lanes and bidirectional drive paths. This new system requires no manually placed traffic rules which makes the designers' job much easier and less time-consuming. An estimate can be used as a feedback to the iterative design process and as a guideline for determining the number of required vehicles. This simulation method is introduced in greater detail in chapter 3.

2.5.2 Analytic methods

While simulation is considered the most reliable method for estimating the system performance of complex AGV systems, several analytical methods have been studied for example by [9], [10] and [11]. Analytic methods provide a quick way to calculate some performance measures at the early stages of decision making. Analytic methods are usually applicable just for simple layout topologies (tandem or single

loop) because it is difficult to consider blocking effects in more complex systems. [10] concludes that analytical methods mostly overestimate system performance. Special care should be taken when these methods are used, and the users should be aware of the strengths and weaknesses when the results are interpreted. Rocla uses Excel based analytic tool as a part of the sales process.

Use of analytical methods is considered useful for making the first estimates about the system performance. Analytical methods usually have quick setup time which means that they could be used during the sales process when the risk of allocating a system engineer to set up a simulation is too big. Obviously required time depends on the analytical method used. For simple formulas it might take very little time to calculate, while more complex models such as CAN-Q [12] have also been developed.

As an example of analytic performance estimation technique, the formulas developed by Egbelu [10] can be used for calculating the number of required vehicles using expected order count during a work shift, shift length, the distance between pick-up and drop-off stations, vehicle speed and time required for loading and unloading a vehicle.

3. DISCRETE-EVENT SIMULATION METHOD

This chapter introduces a discrete-event simulation method which can quickly estimate throughput of an AGV system with a specified number of vehicles. The only inputs for the simulation are a guide-path layout and a list of orders to be performed.

Performance estimates are calculated by simulating the execution of transportation orders. The transportation orders are simulated using a time-window based routing algorithm which plans vehicle movements and stores their locations at specific moments in time. The performance metrics are calculated from the data produced by the simulator. The routing algorithm supports arbitrarily complex bidirectional layouts and also models path congestion. The simulator uses the same routing algorithm that is used in AGV control system sold by Rocla.

3.1 Simulation events

All orders are executed by first visiting the pick-up station and loading the vehicle. Then the loaded vehicle drives to the drop-off station and drops the load. The order is complete after the vehicle has been unloaded. The simulation completes when all orders on the order list are executed. Following assumptions are made when the simulation is running

1. Vehicles do not break down or stop unnecessarily.
2. Loads can not be transferred from one vehicle to another.
3. One vehicle can carry only one load at a time.

The discrete-event simulation method works by planning routes for AGV in the simulated system. The routes are time-dependent and all previously routed plans

are respected when routing a new one. By respecting the previously routed plans the simulation ensures that there are no conflicts between the planned routes. The time in the simulation advances as the orders are executed one by one.

There are three kinds of plans: pick-up, drop-off and release plans. In a pick-up plan, an empty vehicle drives to a pick-up station and picks up a load. After the vehicle has arrived to the station and the loading time has elapsed, the vehicle is given the drop-off order and it drives to a drop-off station with the load. After unload time has elapsed, the system will check if there are new pick-up orders in the order buffer and gives the next order for the vehicle. If the order buffer is empty, the vehicle is released and it drives to an idle location.

3.2 Routing problem

Routing is the process of finding a drivable route for a vehicle. Routing can be divided into two fundamental problems [4]: does the route exist and is it drivable in the current traffic situation. The route may not exists if the layout is divided into unconnected partitions and the target is on a different partition from source. When the route is found, it might not be drivable if there is some other vehicle that is causing a conflict with the routed vehicle. Additionally, the routed vehicle must be able to drive all elements of the routed plan without causing a deadlock in the system.

A layout is a directed graph of segments and points. A simple example of a directed graph is seen in figure 3.1 while a more realistic layout is later shown in chapter 5. A directed graph is constructed of segments which connect points together. There can be multiple segments between the same points. Modern practical AGV systems have conventionally used unidirectional layouts, possibly with multiple parallel lanes going in opposite directions [1]. Unidirectional layouts are used because collision avoidance is easier to handle in them, and to make simple static routing algorithms applicable. For example, Dijkstra's algorithm [13] can be used to find the shortest path route that is not deadlock free but can still be used with some sort of conflict resolving. However, sometimes there is no room for multiple lanes and bidirectional paths must be used. In this case, conflict resolving and traffic control become more difficult [1], [2], and for this reason bidirectional layouts are rarely implemented. One solution to handling conflicts in bidirectional layouts is to use time-window based routing algorithm that avoids conflicts entirely.

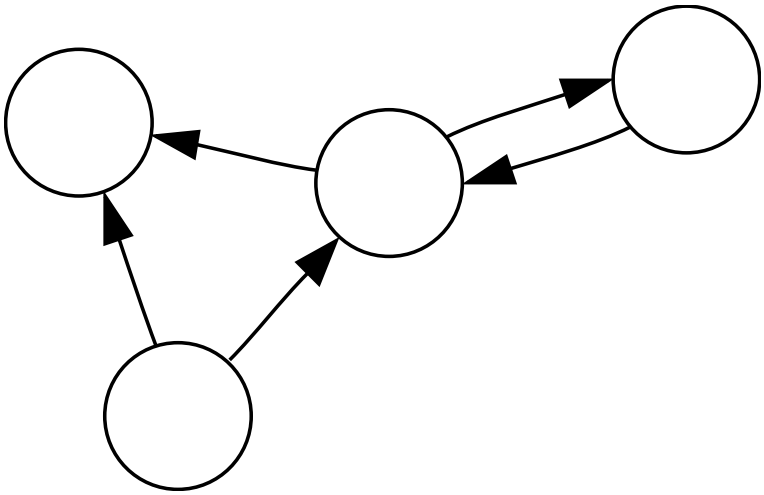

Figure 3.1 *A directed graph*

Static routing is routing without considering time dimension. Dijkstra's algorithm is a classical solution for finding the shortest route from a graph. It works by keeping track of distances to reached points and labeling them as open or closed. First it adds the start point to the closed set and marks its distance as zero. Then the endpoints of the outgoing segments of the start point are added to open set with the segment drive time. If the target point is in the closed set, the shortest path has been found. Otherwise the algorithm picks a point with smallest drive time from the open set. Again, the endpoints of the segments outgoing from the selected point are added to the open set (unless they are found from the closed set). The value of next point is sum of the value of current point and the drive time of the outgoing segment. If some of the points are already in the open set, the drive time value is updated to a smaller value of the two. When all outgoing segments for the current point have been processed, the current point is labeled as closed. Next the algorithm picks a new point with the lowest value from the open set if the target is not found from the closed set. The route can be constructed by following back pointers (assigned to a segment that was used to enter the point) from target point.

$$f(n) = g(n) + h(n) \tag{3.1}$$

There are algorithms which can improve the runtime of graph search compared to Dijkstra's algorithm. For example, A* (pronounced "A-star") [14] is a generic graph search algorithm that improves the runtime by adding a heuristic to plain Dijkstra's algorithm. A heuristic can reduce the number of elements that must be visited until

the target is found. When A*-algorithm picks a point from the open set, it evaluates the cost $f(n)$ using equation 3.1. $g(n)$ is the drive time of element n and $h(n)$ is a heuristic function. The heuristic function should not overestimate the value of $g(n)$ if the lowest cost path is required result. Common and simple heuristic function can be created by pre-calculating distances between all points in the layout. In this heuristic function $h(n) = \text{distance}(target, n)$, where distance function returns precalculated distance between target and n.

3.3 Routing with time windows

Möhring, Köhler, Gawrilow, *et al.* [6] present a time-window based routing algorithm that finds the shortest time route. The shortest time route may differ from the shortest distance route. The route has no collisions with previously routed vehicles. All deadlock situations can also be entirely avoided by using time-window based routing. This allows much more flexible layout design as there is no need to use topology where collision avoidance is easy. The deadlock avoidance and support for arbitrarily complex guide-path layouts was one of the main reasons why Rocla chose this kind of routing algorithm. The use of time-windows also makes it possible to speed up the simulation.

The basic idea behind the time-window based routing algorithm is to create reservations for every visited layout element and its blockings. Following routings can then check if the element has free time-window that is long enough at the time the vehicle would be able arrive to that element. Following constraints are set to guarantee that a route is always found.

1. Routed vehicle must follow segments in the layout until it reaches the target point.
2. Vehicles must not make jumps in time.
3. To consider an element free at moment t, a free time-window starting from t must be long enough for traversing the whole element in minimum time.
4. For element e' to be reachable from element e, a free time window to enter e' should start earlier than last possible exit time on e.

5. Layout must contain idle location points (at least the number of vehicles) that can be occupied without blocking the route from other idle points to activity stations.

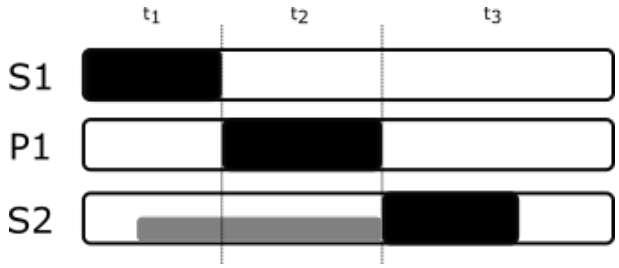

Figure 3.2 *Timelines of elements S1, P1 and S2*

The algorithm works by following free time windows from current element to outgoing elements and selecting the next element to investigate based on the current cost of selected path and precalculated shortest drive time heuristic between all points in the layout. Required computation time does not increase exponentially [6] because all partial plans to a free time window on some element can be expanded by waiting on that element until the end of a free time window.

Reservations that are generated to visited elements are made using primary and secondary reservations. When a vehicle enters an element e it will make a primary reservation on e and secondary reservations on all elements blocked by e. Vehicles must not make overlapping primary - primary or primary - secondary reservations to avoid collisions and deadlocks. Overlapping secondary - secondary reservations are allowed as the elements are not blocking each other.

In figure 3.2 this is illustrated with an example plan of vehicle A going through elements S1, P1 and S2. The black bars are primary reservations of vehicle A and the gray bar on S2 is secondary reservation of vehicle B. Vehicle B is driving some element that blocks S2 so a secondary reservation is generated. Vehicle A cannot enter S2 until the gray reservation on S2 ends, so vehicle A stops on P1 and waits until the gray secondary reservation ends. After that vehicle A continues driving to S2. If there were no vehicle B, vehicle A could drive through P1 without waiting and the black bar would have width of zero.

One of the constraints set previously was the existence of idle locations. These locations are used when there are no orders available but they also have another important use. Whenever a vehicle is routed to some activity station also a secondary

plan, called escape plan, is generated to free idle location. Reservation on escape point extends from virtual arrival time to infinity. The escape plan ensures that vehicle can always find a route to any other point on the layout by following the escape route to escape point and waiting there until a free path opens. The escape plan also releases the reservation on activity station so that other vehicles can find new route there. Escape points must be selected so that they do not interfere with other vehicles even when the vehicle would stand there infinitely. Escape plans are never driven by the vehicles. When there are no more orders to execute, a new release plan is routed for the vehicle.

The job of an order scheduler is to select which vehicle executes a specific order. Connecting a vehicle to an order is called order dispatching. All orders are known when the simulation is started. In a simple case where only a few vehicles and orders are involved, a commonly used scheduling strategy is first-come-first-served (FCFS) [4] strategy. FCFS strategy simply dispatches the order to the nearest idle vehicle. The nearest vehicle can be determined using shortest-path graph search on the layout. The routing algorithm does not impose scheduling constraints, so the FCFS strategy was selected for performance analysis due to its simplicity. Rescheduling of orders is not required as there are no malfunctions or other unexpected disruptions in service.

Many studies list implementation of traffic control as major challenge if bidirectional layout is used. When the time-window based routing is used with drive turns, the control is simple as the routing algorithm guarantees that there will be no deadlocks even when using bidirectional and arbitrary layout topologies.

3.4 Simulation inputs

In the old system the performance analysis required the system engineer to create layout and traffic rules for it. The process is time-consuming because the traffic rules require tweaking whenever the vehicle count in the system is changed. This makes it hard to create experiments with different vehicle counts. Also, quick iteration of different layouts slows down as the engineer must also fix the traffic rules each time.

Time-window based routing algorithm frees the engineer from creating the traffic rules. This allows the quick iteration and experimentation with different layouts and vehicle counts. Orders can be dispatched to vehicles in any order and the system is

guaranteed not to run into deadlocks.

The analysis method requires three input data entities: a layout, an order list and vehicle count. The layout defines how vehicles move around a warehouse. It is composed of layout elements: segments and points. A segment connects two points together and can be traversed only in one direction. However, specifying another segment to opposite direction is allowed. Because vehicles have physical dimensions, the layout also has information about blockings between layout elements. An element holds a list of other elements that can not be used by other vehicles while one vehicle is driving on the element. The list of blocked elements is calculated based on physical vehicle dimensions by collecting all elements under the area that the vehicle sweeps along the element.

Some points in the layout can be used as pick-up and drop-off stations. Idle locations for vehicles that are not executing any order must also be specified. The analysis is based on a time-window based routing algorithm and thus segment traverse times must be known for calculating required time-windows. There are no any additional topological requirements for the layout.

The analysis also requires information about material flow between different areas of the warehouse. The volume of material flow is expressed as a list of transportation orders. One order contains the following information: start time of order, pick-up station, drop-off station and the time it takes to perform the pick-up or drop off activity at the station.

3.5 Simulation outputs

Collecting data for performance metric calculation is done in two steps. The first step is to initialize simulation with layout, orders and vehicles. Vehicles are created to idle locations to be ready to receive pick-up orders. The second step is to run simulation where vehicles execute all orders defined on the order list. The algorithm that the simulator uses for executing the orders is explained in section 4.3.2.

Generated plans can be used to calculate performance metrics like utilization, congestion and system throughput. For calculating the metrics, the reservations of routed plans are saved with information about what kind of plan the vehicle is executing. This data can later be used to analyze where the vehicles are moving and what kind of plan they are executing at any given time.

When the discrete-event simulation has finished, the next step is to calculate performance metrics from generated data. As discussed in section 2.4 there are multiple interesting metrics commonly used to measure AGV system performance. The calculation is introduced in following paragraphs.

A plan $\boldsymbol{P}$ is ordered list of items that have attributes $\boldsymbol{k}$, $\boldsymbol{e}$ and $\boldsymbol{\tau}$, where $\boldsymbol{k}$ is the kind of plan (pick, drop or release), $\boldsymbol{e}$ is identifier for single element in layout (either point or segment) and $\boldsymbol{\tau}$ is the reserved time window $[t_{entry}, t_{exit})$ on $\boldsymbol{e}$.

$\lambda(\boldsymbol{P})$ is a function that returns the drive time of full plan $\boldsymbol{P}$ by summing the length of all time windows in a plan. The length of a time window can be calculated for each element simply by subtracting the t_{entry} from t_{exit}. The load handling time is included to the last time window of the plan.

The system throughput is the number of orders completed in a unit of time. It can be calculated by dividing the total order count with total execution time. Equation 3.2 expresses system throughput as in transports per hour.

$$\text{system throughput} = \frac{\text{order count}}{\text{total time}} \tag{3.2}$$

System utilization (U) and efficiency (E) [7] can be calculated from equations 3.5 and 3.6. T_{empty}, T_{loaded} and T_{idle} are calculated by summing the drive time of pick-up, drop-off and release plans respectively (equations 3.3). Total drive time T (equation 3.4) is calculated by summing all drive times.

$$T_{empty} = \sum_{P \in \mathbb{P}_{pick}} \lambda(P_i) \qquad T_{loaded} = \sum_{P \in \mathbb{P}_{drop}} \lambda(P_i) \qquad T_{idle} = \sum_{P \in \mathbb{P}_{release}} \lambda(P_i) \tag{3.3}$$

$$T = T_{empty} + T_{loaded} + T_{idle} \tag{3.4}$$

Utilization indicates how much time is spent driving plans that contribute completion of orders. This metric is highly affected by the number of awaiting orders in the system. The value decreases if there are unnecessarily high vehicle count compared to order count and dispatch frequency.

Efficiency measures what portion of total drive time is used for transferring loads between activity stations. High efficiency means that more time is spent doing actual work so it is desirable for efficiency to be as high as possible. However, it should be noted that if the system is congested, these measures can become distorted because the vehicles are standing still and waiting for other vehicles, effectively doing nothing.

$$U = \frac{T_{empty} + T_{loaded}}{T} \tag{3.5}$$

$$E = \frac{T_{loaded}}{T} \tag{3.6}$$

Estimation of congestion has been largely ignored or at best approximated in previously published research. The time window based algorithm used for routing gives an accurate value for all vehicle movements in the system. This means that path congestion can be calculated precisely. The routing algorithm is designed so that vehicles wait for other vehicles on points if they can not reach target faster by driving some other route. The time that was used for waiting others can be easily calculated by summing the length of all time-windows on points in pick-up and drop-off plans and subtracting the activity times. If set $\mathbb{W} = \mathbb{P}_{empty} \cup \mathbb{P}_{loaded}$, then

$$\text{blocked} = \frac{\sum_{P \in \mathbb{W}} waittime(P)}{T_{empty} + T_{loaded}}, \quad \text{where} \tag{3.7}$$

waittime is a function that returns time standing on some point in plan P. Blocked time indicates what percentage of utilization time vehicles are waiting for other vehicles. Waiting in release plans is ignored as there is no real difference where the vehicle spends its time; driving, waiting or standing on an idle station. The blockage time is expected to increase the more there are vehicles executing orders in the system.

According to Beamon [7] previous research about determining system performance has often used only one measure to determine systems performance. When multiple measures have been used, no pragmatic method for actual evaluation and interaction of different measures has been analyzed. Generally, [7] suggests that

multiple performance measures should be used in system design to achieve a better understanding of system performance as a whole. A good example of this is how the system efficiency may increase if there is congestion problem on areas where vehicles drive loaded.

4. PROOF OF CONCEPT IMPLEMENTATION

To evaluate the performance and accuracy of discrete-event simulation model, a proof of concept (PoC) implementation was developed. The PoC is heavily based on implementation of actual router used for real system that controls automated vehicles in production environments. Most of modifications were to bypass communication with external world and only depend on the input data given when simulation is launched.

4.1 Architecture and data flow

The PoC uses pipeline architecture consisting of two major processing steps: simulation and analysis. These steps are illustrated in figure 4.1. The first block on left represents the configuration step where order list, layout and other configuration parameters are set. The second block is the simulation program that produces detailed data about vehicle movements. This data is stored for later analysis. In the final block the data is read from store and performance metrics are calculated. It is also possible to generate a visually appealing animation of the simulation where the movement of vehicles is visualized.

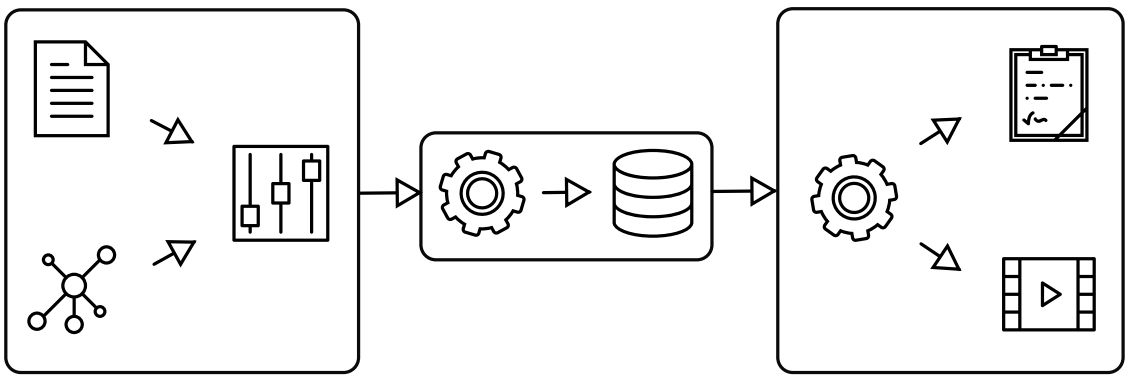

Figure 4.1 *Data flow: configuration, simulation, reporting and visualization*

The simulation output is entirely dependent on inputs and no external state is used. This allows easy parallelization and distributed execution of differently configured simulations.

4.2 Technologies used

The layout is read from a JSON [15] formatted text file. JSON can express data structures such as collection of key-value pairs and lists of values in human and machine-readable format.

The transportation orders are read from CSV [16] formatted text file. Simulator writes vehicle movements to CSV file. CSV format is text-based file format where the first row of a file lists column names separated by a separator character, semi-colon (;) for example, and the following rows list values for all columns, also separated by the separator character.

Routing algorithm and simulation logic are written in F#. F# is a statically typed multiparadigm programming language developed by Microsoft. The implementation uses functional programming style. The simulator was tested on Windows 10 but it should work on any platform supporting .NET runtime.

System performance report is created by querying the data using Structured Query Language (SQL). The data is stored to SQLite database. SQLite was chosen because it required no server installation and the data was easy to transfer between computers in a single file.

Visualization component was implemented using web technologies including HTML5, CSS, JavaScript and WebGL. WebGL is used via three.js library. WebGL based real-time visualization library was required because the animation of vehicle movements was part of the user experience that Rocla wanted to demonstrate in the web application.

4.3 Implementation details

As stated earlier, the implementation of the routing algorithm already existed when the idea about implementing a new discrete-event based simulation method was formed. What had to be done was the control logic that would load the configuration parameters, run the simulation and output the results. The three following sections will specify how these steps were implemented.

4.3.1 Configuration

The design tool for drawing and configuring the layout already existed so there was no need to implement a new one. The design tool allows a designer to draw points and segments, place all types of stations and check how drawn routes block each other. It also allows the user to configure physical dimensions of the vehicles that are used in operations. The layout can be exported from the design tool as a single file which is then loaded into the simulator software. The layout is represented internally as an object that contains three lists, one for all segments, points and stations. Segment object is defined with following properties: identification number (ID), blocked elements as a list of tuples (element type, ID), start point ID, endpoint ID, travel time in milliseconds and segment shape as a list of (x,y)-coordinates. A point object is defined with following properties: ID and blocked elements. Station object is defined with following properties: ID, point ID and station type (activity or idle).

The transportation orders which the simulator will execute are stored into a CSV file that has following columns: start time in seconds relative to first order on the list, pick-up station ID, drop-off station ID and the time it takes to perform the operation (pick-up/drop-off) at the stations. Additionally, the user can specify the number of vehicles in the system when the simulation is started.

Figure 4.2 *The routing order of the vehicles*

4.3.2 Discrete-event simulation

The simulation starts when all configuration variables are loaded and stored into internal data structures. The current state of the simulation will be stored in a variable called world and it will hold all reservations of the routed vehicles.

The simulation runs in a loop. The loop terminates when all transportation orders are completed. The first step inside the loop is to select the vehicle with earliest plan finish time in the world. This selection is illustrated in figure 4.2. The figure represents the reservations for each vehicle (rows) over time (x-axis). When the vehicles are routed based on the earliest plan finish time, the world will always contain all blocking reservations that affect the routing result.

When the vehicle is routed, the reservations of its old plan are released from the world. This is an optimization to keep the data structure from becoming too large and therefore slow down later routings. As an example, in routing number 6, the vehicle is constrained by reservations created by routings 2, 3 and 5. The reservations of routing 1 have already been released by routing 5, and the routings 7 and 8 have not yet happened.

So, the vehicle that has the earliest plan finish time must be routed first. Next, the simulator determines appropriate routing command based on the current state of the selected vehicle. If it has already picked a load, the vehicle must be routed to the drop-off station of the order. Otherwise the simulator assigns the next transportation order for the vehicle from the order list. If there are no awaiting pick-up orders, the vehicle is routed to an idle station. There the vehicle can wait for new order without interfering other vehicles. After the routing has completed, the world is updated with the new plan reservations and loop termination condition is checked.

After each routing, the simulator writes vehicle movements to a file. The file contains a row for each reservation along the path that the vehicle was routed through. The reservations are stored using following columns: order ID, vehicle ID, order stage, element type, element ID, element start time and element finish time.

4.3.3 Reporting and visualization

A performance report is generated using the data from previous simulation step. The data is first imported into a database and then SQL is used to calculate the performance metrics for different vehicle counts. Calculated metrics are outputted in JSON format and saved to file. The file contains following data:

- number of vehicles in the system
- total time for completing the order list in hours
- transports per hour (equation 3.2)
- time used for load handling as percentage of total time
- time the vehicles were blocked as percentage of total time
- time the vehicles were driving empty as percentage of total time
- time the vehicles were driving loaded as percentage of total time

If the simulation is run with multiple different vehicle counts, the performance report is generated for all of them and the results are plotted on graph automatically. The graphs make it easy to visually inspect how different vehicle counts perform comparatively.

Visualization draws the layout (example is seen in figure 5.1) and animates vehicle movements over time. The viewer can accelerate the animation from real-time speed. Implementation reads the CSV file from simulation and builds an internal data structure for querying vehicle locations on a specific point in time.

The animation is useful for visual analyzation of system performance and for identifying obvious choke points on the layout. It is also helpful as a debugging and verification tool because the developer can immediately see if some vehicle disappears, makes an unexpected jump or collides with some other vehicle. These observations were one of the main methods for testing and evaluating whether the PoC implementation was working correctly.

4.4 Simplifications

The PoC implementation is not completely realistic simulation of the real-world system. It does not support all commonly used features such as different scheduling algorithms or dynamic obstacle avoidance.

The simulation is mainly focused on modeling the behavior of time-window based routing algorithm that avoids deadlocks in system. In addition to path congestion, there are many other factors that affect the performance of AGV system. For example, vehicles are commonly powered with (re)chargeable batteries. The charging and changing of batteries must be managed so that the system does not run out of energy. The PoC implementation does not simulate battery consumption and all vehicles are expected to have infinite battery life.

Current implementation lacks support for multiple vehicle types. It is common that not all tasks are possible with all vehicle types. For example, a paper roll needs a clamp like lifting arm while standard pallets are moved around using forks. For this reason, the simulator should support different vehicle types.

Order dispatching and scheduling decide which vehicle is allocated for a specific order and which orders are completed first. The current implementation uses a simple algorithm that executes all dispatches the orders sequentially and selects the first available vehicle to perform the order. Additionally, the current implementation can not switch orders from one vehicle to another after the order is dispatched to a vehicle.

Vehicles are expected to follow planned routes perfectly. In real-world there are always unexpected breakdowns or other unplanned delays. These may be caused by external path blockages such as humans or manually controlled vehicles. These kinds of unexpected events are not currently simulated.

Finally, the simulator does not manage storage location capacity. For this reason, it is possible to drop an infinite amount of loads to a single drop-off station. Managing the storage room is left to one of the responsibilities of order list generator. In real-world there are also differences in the required operation time when the storage location holds a different number of items. For example, when pallets are stacked it takes longer to lift new pallet on top of the stack if there are already a large number of them in storage.

5. EVALUATION

One of the recognized issues at Rocla is the lack of system performance estimation methods which are usable in the early stages of the sales process. The continuous simulation method used for system performance analysis suffers from a long simulation time. It also has relatively high skill requirements for setting it up, which means that the sales can not perform the simulations without a help from the design engineering team.

This chapter evaluates the results obtained from the discrete-event simulation method and compares them to the results from the continuous simulation method. The runtime performance of the discrete-event simulation method is inspected and the extendability and ease of use are considered. Further development ideas are also discussed. Final section shares some thoughts about how the project went and how well it establishes the goals set for this thesis project.

5.1 Results

The simulation was run using both continuous (CS) and discrete-event simulations (DES). Multiple vehicle counts were simulated in both simulators in order to compare the results. A low number of vehicles (1 and 2) were not simulated using CS because of the time requirements of such simulation. However, the results from DES give some perspective to the required time when using such small vehicle counts. All simulations were run on a layout shown in figure 5.1. The transportation order list included 112 transportation orders, requiring the vehicles to transport virtual loads between pick-up and drop-off stations around the layout.

Figure 5.2 represents system throughput from continuous and discrete-event simulations. The graph shows that a single vehicle could reach a throughput rate of 11.7 orders per hour. As more vehicles were added to the system, the throughput increased almost linearly when no more than eight vehicles were used in the system.

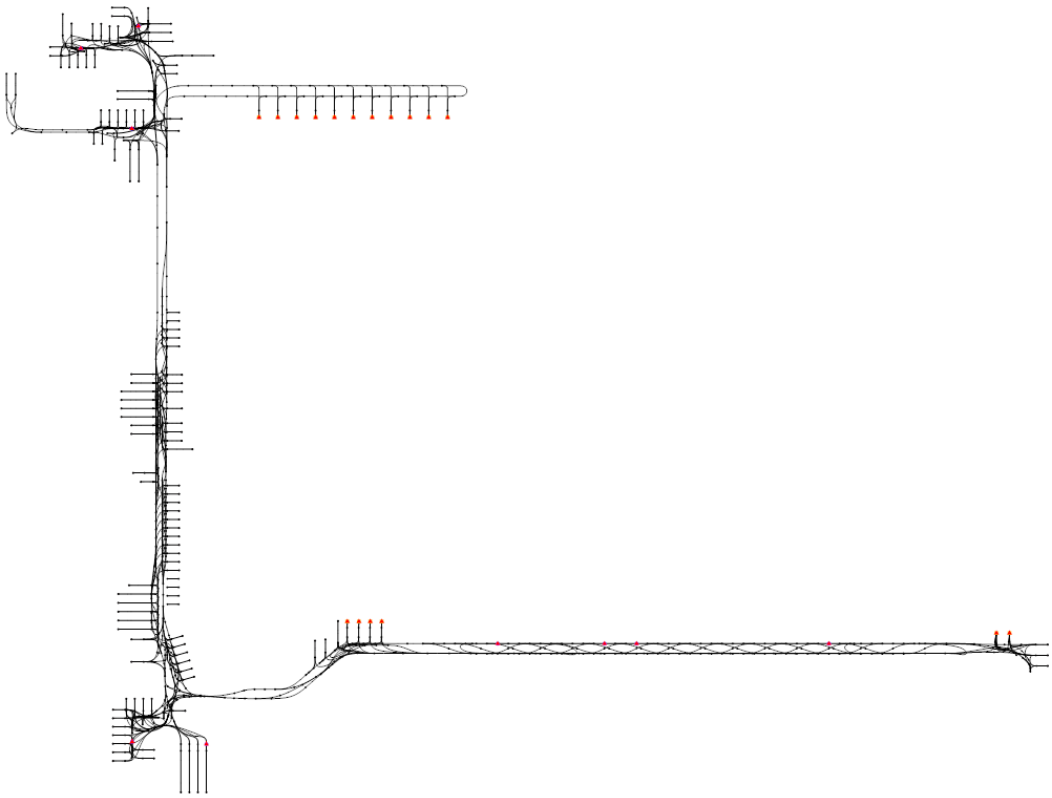

Figure 5.1 *Simulation layout*

Twelve vehicles were capable to perform 68.6 orders per hour but the increase from 62.8 achieved by eight vehicles is relatively small.

The wall clock time required for executing all 112 orders in the transportation order list can be seen from figure 5.3. The durations not shown in the figure were for a single vehicle: 9 hours and 35 minutes and for two vehicles: 5 hours and 3 minutes.

The accuracy of the new method was one of the main requirements as there is little use for the results which do not match up with the real world. Based on the results seen in figure 5.2, the discrete-event simulation method seems to overestimate the performance. However, the offset seems to be fairly constant so it could be compensated when the results are analyzed. A larger result data set is required to determine if the offset really is constant.

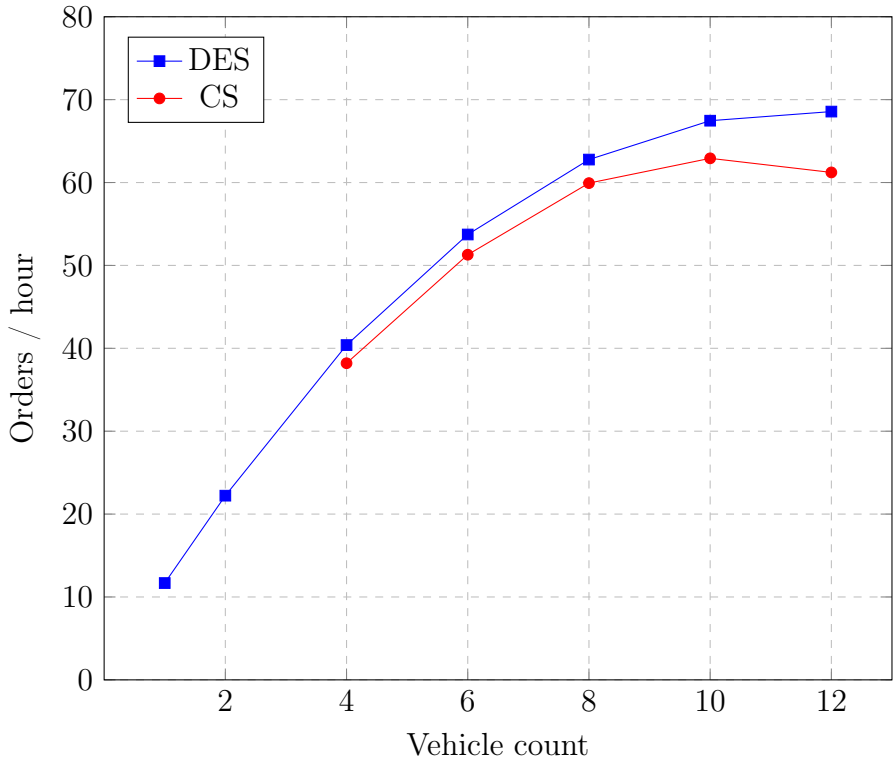

Figure 5.2 *System throughput*

Based on the experiences at Rocla, order dispatching affects to the throughput of the system. For example, if a simple nearest vehicle-rule is used, just a small delay in the completion of an order can lead to a situation where a vehicle must drive a long distance to next pick location, even though another vehicle would have been free just after a few seconds right next to the pick location. If the scheduling algorithm was smart enough, it could detect this and delay the order until the vehicle next to the pick location is released.

The PoC implementation uses very simple order dispatch strategy which assigns the orders to a free vehicle using first-in-first-out algorithm. More efficient order dispatching algorithms are available and could be utilized in discrete-event simulation method. The continuous simulation method already supports different strategies. Using the same strategy in the discrete-event simulation method could produce more similar results with continuous simulation method.

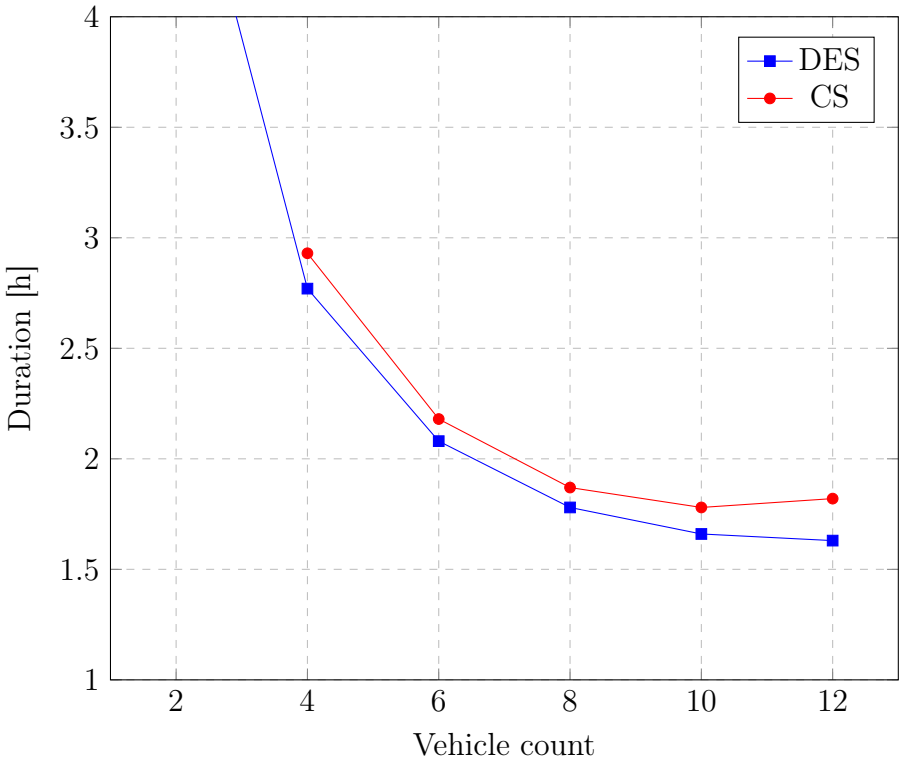

Figure 5.3 *Order list execution time*

Figure 5.4 shows how the path congestion increases as more vehicles are added to the system. When twelve vehicles were used, the blockage percentage reached 45%. That means almost half of the time was spent standing still waiting for other vehicles. In contrast, when two or four vehicles were used, the blockage percentage stayed under 10%.

While the results given by continuous simulation method slightly deviate between rounds the results given by the discrete-event simulation method are completely deterministic. The deviations in the results of the continuous simulation method are caused partly by optimizations performed by the runtime and partly because of randomness in the execution environment. The continuous simulator also uses different order dispatching algorithm than the discrete-event simulator.

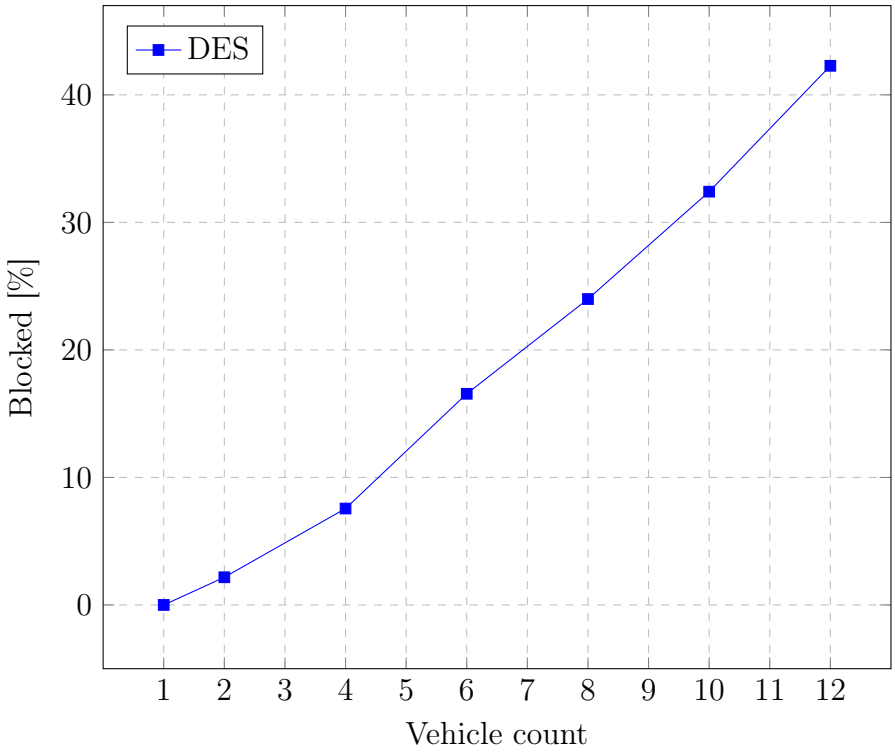

Figure 5.4 *Blocked percentage*

5.2 Runtime performance

The runtime performance of simulation methods is measured as the time required for obtaining the performance metrics. The process can be split into three different portions: setup time, simulation time and analysis time. The runtime performance is a very important factor because if the simulation takes a really long time to complete, it can not be used for all different use cases such as layout optimization.

The old vehicle control system that did not use time-window based router required the designer to define traffic rules to the layout. This increased the setup time considerably and required a lot of skill from the person who drew the layout. In practice that meant that it was not financially feasible to perform the analysis in the early stages of the sales process. When the new time-window based router became available for the design engineers they were able to draw the layout without manually placed traffic rules. This meant that they had more options in testing the system with different vehicle counts without altering the layout.

The runtime performance is the most important improvement of discrete-event simulation method compared to the continuous simulation method. The simulation

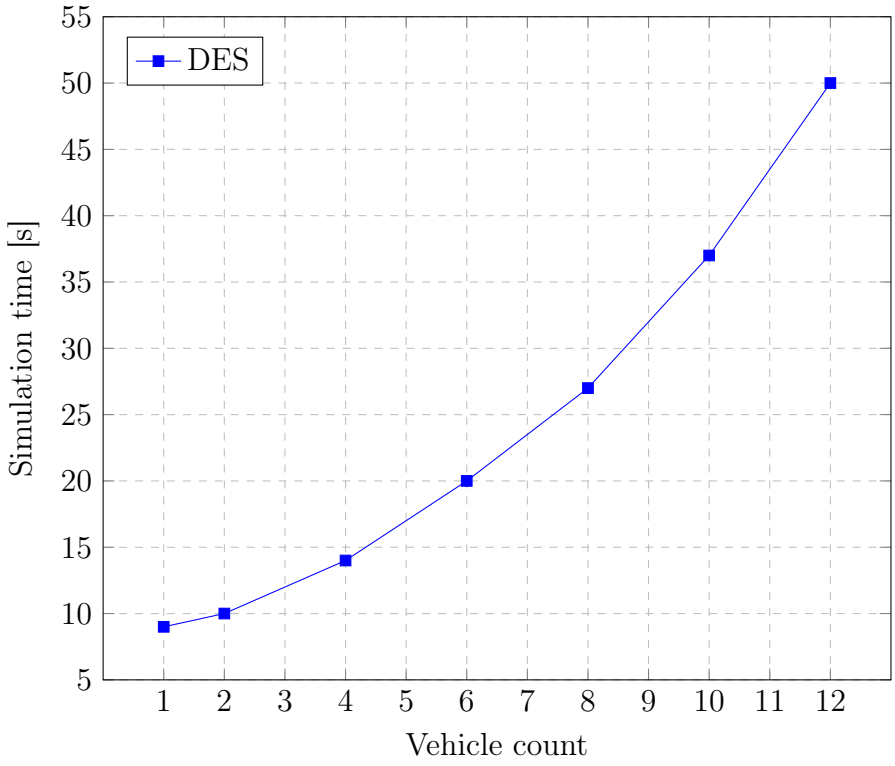

Figure 5.5 *Runtime performance*

time of discrete-event simulation method can be seen from the figure 5.5. The PoC implementation can produce almost the same performance measures as the CS method simulation, while decreasing the simulation time to a few dozen of seconds. The figure also shows how the runtime increases when more vehicles are simulated. The reason for this effect is the time complexity of the time-window based routing algorithm which is dependent on the number of vehicles. The overall runtime performance improvement compared to continuous simulation method when using 12 vehicles is over a magnitude.

5.3 Extendability

The simulator should be able to adapt to constant changes in factory layouts or new vehicle models. For example, when a new vehicle model is released, it most likely has different physical dimensions or faster drive motor. In current implementation, these properties are completely isolated from the simulation runtime and are statically specified in the layout file. When the layout is generated by the layout design software, it calculates the blockings between elements based on the physical

dimensions of the vehicle. Segment drive times are also calculated based on the configured max speed of the vehicle. The design software also calculates slower drive time for segments with tight turns.

A system designer might also be interested to try different order scheduling and dispatching strategies. However, currently there is no option to change this dynamically, but it should be possible to create a set of predefined strategies that the designer can choose from. That way it would be possible to evaluate which strategy works best for the specific vehicle and layout configuration.

5.4 Ease of use

To use the simulator, the user must first design and draw the routing layout. Drawing the layout is the most time-consuming task of the simulation process although it has become considerably more manageable as the designer no longer needs to configure traffic rules for the layout. If the user wants to simulate on a layout of an already existing system, the layout requires no additional configuration for it to be usable. Drawing the layout is out of the scope of this thesis so it is not discussed further.

In addition to system layout, the user must obtain an order list that specifies the transportation tasks. Generating the order list can be done using an existing tool which generates random transportation orders between stations or by hand based on the expected or measured material flows in the simulated system.

The simulator itself is fairly easy to use. When inputs are ready, the user selects the number of vehicles. Multiple vehicle counts can be selected at the same time. The simulator will start parallel simulations for each selected vehicle count. After the simulation is started, the user has to wait until the simulation finishes. An estimate about remaining time is displayed while the simulation is running. The PoC implementation automatically calculates the performance metrics (see 4.3.3) and draws charts with selected vehicle counts.

When the results are ready the user must analyze and interpret performance charts. This requires knowledge and experience from the user as there is no "correct answer" and desired system properties can be weighted differently in different systems.

5.5 Further development

Because time-window based routing algorithm generates accurate time windows for all transportation tasks, the planned routes could be used for identifying problematic areas in the layout. For example, the points in the layout can be ranked by waiting time or the segments can be ranked by how many reservations they have. This could reveal which sections of the layout are bottlenecks in current design and which ones are used less or never.

As discussed in previously, the layout design is one of the major costs in system design. Currently the routing map must be drawn manually using a design software. The salesmen should be able to create an initial layout for the simulator automatically. By automating the layout generation, whole estimation process could be automated. The automated layout generation could work by first marking the warehouse hallways and activity station locations to blueprints of the warehouse and then using a path generation algorithm to find routes between activity stations.

5.6 Experience

Implementation of the discrete-event simulator runtime was fairly straightforward as the time-window based router was already available. The router required only slight modifications to be usable for the discrete-event simulation. Additional code was written for passing the inputs to the algorithm and generating the output file which contains the planned routes. Storing these routes to the database for analysis was a simple task. Writing SQL queries for desired performance metrics required a bit more work but was not too hard. Plotting the results could be done using Microsoft Excel.

The visualization was really useful in debugging as you could very easily recognize if there was some error in the designed plans. For example, if two vehicles disappeared or drove through each other it was a clear indication that there was a bug in the simulator. The visualization is also a very convincing way to demonstrate how the simulator works to the customer.

6. CONCLUSIONS

The main motivation for writing this thesis raised from a real world need - a need to have a quick and easy tool for estimating AGV system performance. The project goal was to develop a new, faster method that could be used instead of old and slow continuous simulation (CS) method that was used in performance estimation process. The users of the new method should be able to accurately estimate the throughput of AGV systems operating in warehouse and manufacturing environments.

As a result of this project, a new discrete-event simulation (DES) method was designed and a tool for running the simulations was developed. The tool takes a system layout and an order list as inputs and calculates system throughput for requested vehicle counts. The developed simulator can also output animated video for vehicle movements around the layout. The video is a nice way to show customers how the time-window based routing algorithm finds deadlock-free routes for the vehicles, and how it increases the performance of the system by allowing arbitrary bidirectional layouts to be used. The video was also used to verify that the simulator is working properly. The user can immediately see from the video if some vehicles collide or move erratically.

The system performance estimate calculated using the DES method produces similar results as previous CS method. The biggest advantage of DES method is radically reduced simulation time. It also produces data that was previously unavailable. New data can support decision-making process when deciding the required number of vehicles for a specific system. Most important new metric is the blocking factor that indicates how much congestion there is in the system. Blocking factor is not currently available when the CS method is used.

When testing the implemented DES method based simulator, there was a limited capability to run multiple simulations with old continuous simulation method due to its excessive runtime requirements. Also the confidentiality of real world

layouts restricted this study to include only one warehouse AGV system. Further experiments should be performed to evaluate the real accuracy of the new discrete-event simulation model. The support for analyzing the effects of different scheduling strategies should be implemented. Also, in order to support the sales process, the feasibility of automatic layout generation could be investigated and implemented.

After the first PoC implementation was demonstrated to Rocla they decided to implement equivalent functionality to existing debug and design tools. Currently it ships as a part of the latest version of these tools.

BIBLIOGRAPHY

[1] T. Le-Anh and M. B. M. de Koster, "A review of design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 171, no. 1, pp. 1–23, 2006.

[2] I. F. Vis, "Survey of research in the design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 170, no. 3, pp. 677–709, 2006.

[3] Rocla. (2017). Maximise your logistics performance – AGV range brochure, [Online]. Available: `https://www.rocla-agv.com/sites/default/files/rocla_agv_general_brochure_web_pdf.pdf` (visited on 10/17/2017).

[4] L. Qiu, W.-J. Hsu, S.-Y. Huang, and H. Wang, "Scheduling and routing algorithms for AGVs: A survey," *International Journal of Production Research*, vol. 40, no. 3, pp. 745–760, 2002.

[5] S. Rajotia, K. Shanker, and J. Batra, "A semi-dynamic time window constrained routeing strategy in an agv system," *International Journal of Production Research*, vol. 36, no. 1, pp. 35–50, 1998.

[6] R. H. Möhring, E. Köhler, E. Gawrilow, and B. Stenzel, "Operations Research Proceedings 2004: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR). Jointly Organized with the Netherlands Society for Operations Research (NGB) Tilburg, September 1–3, 2004," in. 2005, ch. Conflict-free real-time AGV routing, pp. 18–24.

[7] B. M. Beamon, "Performance, reliability, and performability of material handling systems," *International Journal of Production Research*, vol. 36, no. 2, pp. 377–393, 1998.

[8] J. Banks, *Handbook of simulation: Principles, methodology, advances, applications, and practice*, ser. A Wiley-Interscience publication. Wiley, 1998.

[9] W. Maxwell and J. Muckstadt, "Design of automated guided vehicle systems," *IIE Transactions*, vol. 14, no. 2, pp. 114–124, 1982.

[10] J. Egbelu, "The use of non-simulation approaches in estimating vehicle requirements in an automated guided vehicle based transport system," vol. 4, 1987, pp. 17–32.

[11] D. Sinriech and T. J.M.A., "An economic model for determining AGV fleet size," in, 6. 1992, vol. 30, pp. 1255–1268.

[12] J. Tanchoco, P. Egbelu, and F. Taghaboni, "Determination of the total number of vehicles in an AGV-based material transport system," *Material Flow*, no. 4, pp. 33–51, 1987.

[13] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[14] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. SSC-4(2), pp. 100–107, 1968.

[15] E. T. Bray, "The javascript object notation (json) data interchange format," IETF, RFC 2070-1721, Mar. 2014.

[16] Y. Shafranovich, "Common format and mime type for comma-separated values (csv) files," IETF, RFC 4180, Oct. 2005.